Webサービス連携のためのトランザクションの隔離性について

徐　海燕† 古川哲也†† 史　一華†††

† 福岡工業大学情報工学部　〒 811-0295 福岡市東区和白東 3-30-1
†† 九州大学大学院経済学研究院　〒 812-8581 福岡市東区箱崎 6-19-1
††† 西南学院大学商学部　〒 814-8511 福岡市早良区西新 6–2–92
E-mail: †xu@cs.fit.ac.jp, ††furukawa@en.kyushu-u.ac.jp, †††shi@seinan-gu.ac.jp

あらまし　Web サービスの普及に伴って、インターネット上に公開された様々な Web サービスを構成部品として利用し、企業の情報システムを組み立てるという時代が到来する。このため、各々のトランザクションの並行実行が互いに与えうる影響について検討する必要がある。本論文では、Web サービスに従って単独で実行された各インスタンスが満たさなければならない性質を一貫性、並行実行時に同じ性質を満たすための性質を隔離性と定義し、隔離性を保証するための ACID 属性よりも緩められた条件を提案する。また、BPEL4WS のようなフローエンジンが他のサービスを呼び出す場合における隔離性の実現法について検討する。
キーワード　Web サービス、トランザクション処理、BPEL4WS、隔離性

Isolation Property of Web Service Transactions

Haiyan XU†, Tetsuya FURUKAWA††, and Yihua SHI†††

† Dept. Computer Science and Engineering, Wajirohigashi 3-30-1, Higashi-ku, Fukuoka, 811-0295 Japan
†† Dept. Economic Engineering, Kyushu University, Hakozaki 6-19-1, Higashi-ku, Fukuoka, 812-8581 Japan
††† Faculty of Commerce, Seinan Gakuin University, Nishijin 6–2–92, Sawara-ku, Fukuoka, 814-8511 Japan
E-mail: †xu@cs.fit.ac.jp, ††furukawa@en.kyushu-u.ac.jp, †††shi@seinan-gu.ac.jp

Abstract　Web services have quickly become a very important technology for enterprise system development and it has become possible to build distributed systems using web services as components. This paper discusses the isolation property of web service transactions. By classifying the input and output data of each primitive activity, we propose a condition to guarantee the isolation property in web service transactions, which is weaker than the conventional serializablity property. The mechanisms for the isolation are also discussed.
Key words　Web service, transaction processing, BPEL4WS, isolation

1. はじめに

ビジネス取引の自動化を目指すために、Web サービスについての研究が盛んに行われている。複数の Web サービスをインターネット上で統合して新たな Web サービスを定義するために、2001 年に WSFL と XLANG が発表され、2002 年にその両者を融合した BPEL4WS (Business Process Execution Language for Web Services) が発表された [1]。BPEL4WS によるビジネス記述では、BPEL4WS が主体となって他のサービスを呼び出す形式を採用しており、$< invoke >, < receive >, < reply >$ といった基本的な Web サービスを通して他のサービス（パートナー）とメッセージのやり取りを行う。さらに、Web サービス間の流れは、受信したメッセージによって分岐したり合流したりというように定義することができる。一方、W3C で標準化が行われている WSCI (Web Services Choreography Interface) や WSCL (Web Services Conversation Language) もよく知られている。

複合 Web サービスの設計と分析に際しては、どのような複合 Web サービスは正しく動くのか、どのように Web サービスの実行の正しさを保証すればよいのか、などのように、多くの課題が残されている [4]。インターネット上に公開された様々な Web サービスを構成部品として利用し、企業の情報システムを組み立てていくための研究も活発に行われている [7]。本論文では、Web サービスの密結合によって企業内の情報システムを組み立てる際に必要となるトランザクションの隔離性の管理について検討し、一般的な疎結合の場合まで触れる。

密結合による企業内の情報システムにおいて、複数のインスタンスが同時に実行されるとき、トランザクションの隔離性の

管理が必要となる場合がある。

［**例 1**］ 融資の申込みを処理するプロセスが図 1 のように構築されているとする。顧客の融資申込みを受け付け、その顧客に関する収入／財産／ローンなどの経済状況をデータベースから検索し、金融機関の Web サービスであるローン審査を呼び出してその顧客の融資へのリスクを判断する。融資のリスクが低いと判断された場合は、融資を承認し、ローンの貸出しを行う。それ以外の場合は、融資ができない旨の返信を行う。しかし、ある顧客が複数件の融資申請を同時に行った場合、貸出しできないローンまで貸し出してしまう場合が生じうる。 □

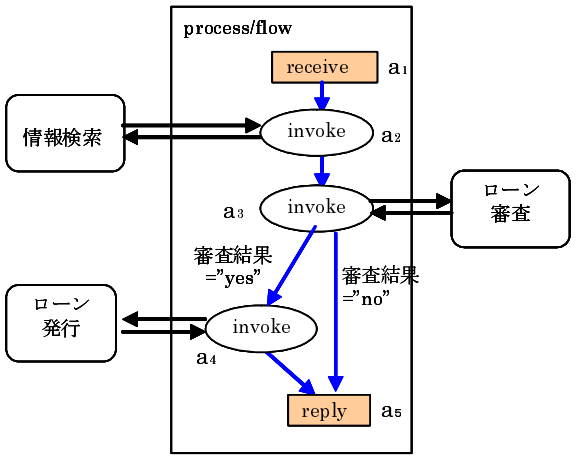

図 1　融資申込みプロセス

従来のトランザクション管理は、原子性（Atomicity）、一貫性（Consistency）、隔離性（Isolation）、耐久性（Durability）によって表される ACID 属性に基づいている [2]。しかし、この ACID 属性は Web サービスのような疎結合や密結合がある結合環境においては、柔軟性に欠け、制限が多すぎるという問題が広く認識されている [5]。このため、標準化組織 OASIS による Business Transaction Processing (BTP) や、BEA、IBM、Microsoft による WS-Transaction とその付属仕様である WS-Coordination が公開されている。BTP は疎結合システムへの適用を前提としており、WS-Transaction は密結合用の ACID トランザクションと疎結合のための長大トランザクションの両者をサポートしている。これらの仕様に共通しているのは、全て実行されるかまったく実行されないかという後退復帰方式ではなく、ビジネスロジックに合わせた個々の例外処理を行う補償方式を使用することである。しかし、Web サービスの連携によって企業内の情報システムを組み立てるという密結合の場合においても厳密な議論はなされておらず、ACID 属性が必ずしも適しているとは限らない。

Web サービスの例として旅行予約の場合がよく用いられている。例えば、図 2 に示している旅行予約 Web サービスの例では、旅行代理店、カード会社、航空会社の 3 つの企業が関わっている疎結合の場合である。航空会社に関わる切符予約と発券という二つの Web サービスの実行であるトランザクションの隔離性（すなわち、予約が取れれば、発券までに他にどんなトランザクションが実行されても影響を受けることはない）は、サービスの内部ロジックによって実現している。従来の企業に

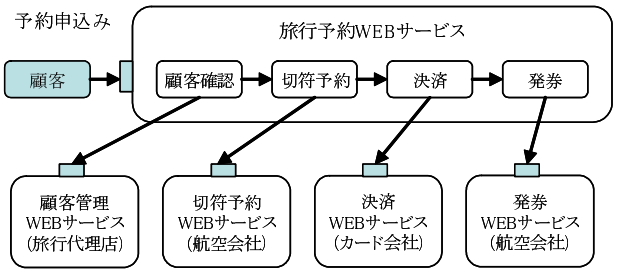

図 2　旅行予約 Web サービス

おけるワークフローシステムにおいても、トランザクションの隔離性が内部設計によって実現されている場合が多い。しかし、Web サービスの発展につれて、インターネット上の様々な Web サービスを用いて新しい Web サービスを組み立てていく時代が到来する。インターネット上で公開している Web サービスを利用する場合に、入手できるのはインタフェースに関する情報のみで内部ロジックに頼ることは現実的でない。本論文では、各々のトランザクションが単独で実行する場合にデータは正しく管理できるという前提の下で、密結合の場合におけるトランザクションの隔離性の性質について検討し、ACID 属性よりも緩められることを示す。さらに、本論文の結論は、密結合である企業内システムを疎結合によって融合していく一般的な場合（図 3）においても適用できることを示す。

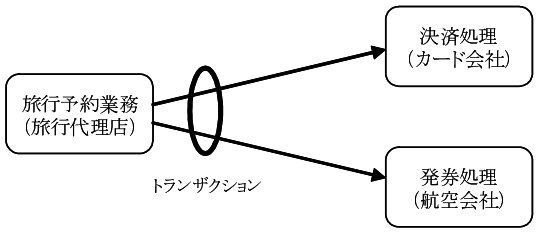

図 3　旅行予約 Web サービスの構成

本論文では、Web サービスにおける基本的な操作単位であるアクティビティの入出力データの種類について分析し、トランザクションにおける一貫性上で関連するデータ項目の集合を明確にする。それに基づいてトランザクション同士が互いに相手に干渉されていないことを保証するための隔離性の性質について検討する。また、入出力データ間の対応関係という意味論を利用できる場合における隔離性の拡張可能性について検討する。さらに、BPEL4WS エンジンのように自身を主体として他のサービスを呼び出すフローエンジンに対して、施錠方式による隔離性の実現方法について議論する。

本論文は、次のように構成される。2 章で Web サービスとその連携であるフローモデルを定義し、3 章では密結合と疎結合の場合に分けて、Web サービスのトランザクションの隔離性の正当性基準について検討する。4 章では BPEL4WS エンジンのようなフローエンジンとデータベースの連携による隔離性の実現方法を、施錠方式の立場から提案する。5 章は全体のまとめである。

2. Web サービスモデル

本章では、Web サービスとその連携によって企業内の情報システムを構築する場合の一般的な記述モデルを与える。

Web サービスは XML フォーマットのメッセージデータを交換する。使用されるメッセージのデータフォーマットや、メッセージの処理ルールは、通信規約 SOAP によって定められている。Web サービスのインタフェースは、WSDL によって記述される。複数の Web サービスを連携して新たな複合 Web サービスを作成する際は、パートナーによって提供される Web サービスを呼び出す $< invoke >$、パートナーからのメッセージを受け取る $< receive >$、パートナーへメッセージを送信する $< reply >$、大域変数の引き渡しを行う $< assign >$ が基本的なアクティビティになる。

個々の基本的なアクティビティのインタフェースを定義する WSDL には、名前、パートナー、ポートタイプ、操作、入出力データといったパラメータについての記述が含まれている。本論文では、基本的なアクティビティ a を $a(I, O, M, p)$ として抽象化して記述する。ここで、I は a によって検索される大域変数、O は a によって値が代入される大域変数である。例えば、基本的なアクティビティでは、$< receive >$ は出力データのみを、$< reply >$ は入力データのみを持つ。$< invoke >$ と $< assign >$ は入力データと出力データの両方を含む。一方、M は O の個々の要素を生成するために用いられている I の要素という O と I の要素間の対応関係を記述しているが、M についての記述が省略される場合は、O の各要素が I のすべての要素と対応しているとする。また、p は a のパートナーを記述しているが、パートナーが重要でない場合は省略する。

I と O の和集合をアクティビティ $a(I, O, M, p)$ の操作データ集合という。実際の場合には、利用される Web サービスによってインタフェースの粒度が異なってくる。一般的に企業内ではメッセージが必要なビジネス情報を全て含むような粒度の細かいインタフェース、外部利用者に対してはキーワード情報しか含まない粒度の粗いインタフェースが使われている。本論文では、企業内の連携においては、必要とする全ての入出力情報をインタフェースに含むとする。

［**例 2**］ 図 1 には、a_1, a_2, a_3, a_4, a_5 の 5 つの基本的なアクティビティがある。それらの記述を次に示す。

- 申請受付 a_1:
  $I_1 = \phi$,
  O_1 =｛顧客 ID:x_1, 申請額:x_2｝,
- 情報検索 a_2:
  I_2 =｛顧客 ID:x_1, 申請額:x_2｝,
  O_2 = ｛経済状況:x_3｝,
- 審査 a_3:
  I_3 =｛顧客 ID:x_1, 申請額:x_2, 経済状況:x_3｝,
  O_3 = ｛審査結果：x_4｝,
- ローン発行 a_4:
  I_4 =｛顧客 ID:x_1, 申請額:x_2｝,
  O_4=｛経済状況:x_3｝
- 結果返信 a_5:
  I_5 =｛審査結果：x_4｝,
  $O_5 = \phi$.

すなわち、a_1 では顧客の申請を受け付け、顧客 ID や申請額を大域変数に代入している。a_2 では企業データベースに記憶されている申請者の経済状況に関する情報を、a_1 で代入された大域変数顧客 ID をキーワード にして検索し、その結果を大域変数 x_3 に代入している。そして、それらによって得られた情報に基づいて、申請者に対する審査を a_3 によって行う。a_4 で申請額のローンを発行するが、申請者の経済状況にも今回のローンの発行を反映する。a_5 では顧客への審査結果の返信を行う。□

各節点の作業によってデータベースに対しても検索や変更がなされているので、I_{DB} と O_{DB} でそれぞれ各節点によって検索と変更がなされたデータベースの項目を表すとする。例 2 においてデータベースに対する入出力データは次のようになる。

I_{2DB} =｛顧客 ID:x_1, 経済状況:x_3｝
I_{4DB} =｛顧客 ID:x_1｝
O_{4DB}=｛経済状況:x_3｝

データベースからの入力データとデータベースへの出力データに関する情報は、アクティビティのインタフェースからは得られない。本論文では、データベースレベルでの介入によってそれらの情報を入手する方法を導入する。

一方、基本的なアクティビティ間の実行順序は、制御フローによって記述する。BPEL4WS では $< sequence >, < flow >$ などの構造化されたアクティビティによって、基本的なアクティビティのインスタンス間の実行順序を定義している（図 1)。一方、WSFL では、controlLink と dataLink で基本的なアクティビティ間の処理の流れとデータの流れを記述している。制御フローの記述方法はいずれにせよ、制御フローに従う実行であるトランザクションが生成される。本論文では、制御フローの記述方法には制限を設けない。

制御フローに従う実行であるトランザクションを記述するために、データ項目 x およびデータ項目集合 X のインスタンスを $\mathcal{I}_x$ および $\mathcal{I}_X$ とし、$\mathcal{A}(\mathcal{I}_I, \mathcal{I}_O, M, p)$ で基本的なアクティビティ $a(I, O, M, p)$ の実行を記述する。トランザクションは、基本的なアクティビティの実行を表す節点と XOR 分岐節点／結合節点からなる節点集合と、それらの間の実行順序を表す枝集合からなる有向グラフ $WT(TN, TE)$ である。XOR 分岐節点に対しては分岐枝は 1 つしか存在せず、真となる経路条件に関わるデータ項目および値は、分岐節点の出力データとそのインスタンス $\mathcal{I}_O$ となる。トランザクション $WT(TN, TE)$ は、有向非巡回グラフである。これは、複数回実行される基本的なアクティビティは異なる節点に展開され、条件分岐は特定の経路のタスクに従って実行されるためである。

トランザクション $WT(TN, TE)$ におけるデータの流れは、通常データフローと呼ばれる有向グラフ $DT(TN, DE)$ によって記述される。節点 $\mathcal{A}_i$ が $WT(TN, TE)$ において祖先である $\mathcal{A}_j$ の出力データを入力データとして利用するなら、データフローに $\mathcal{A}_j$ から $\mathcal{A}_i$ への枝がある。すなわち、データフロー中の枝

はトランザクションの枝の推移的閉包に含まれる ($DE \subseteqq TE^*$)。

異なるトランザクションが同時に実行される場合に、スケジュールが生成される。$WT_i(TN_i, TE_i)$ $(i = 1, 2, \cdots)$ からなるスケジュールは、次のような有向非巡回グラフ $WH(HN, HE)$ である。

- $HN = \bigcup_i TN_i$
- $HE \supseteqq \bigcup_i TE_i$
- 異なるトランザクションの節点 $\mathcal{A}_s$ と $\mathcal{A}_t$ が競合する $((O_{sDB} \cap (I_{tDB} \cup O_{tDB}) \neq \phi) \vee (O_{tDB} \cap (I_{sDB} \cup O_{sDB}) \neq \phi))$ なら、 それらの間には実行順序を示す枝か推移的な枝が HE に含まれる。

[**例 3**] 図 4 は例 1 の Web サービスに従う実行 WH_1 を示している。 □

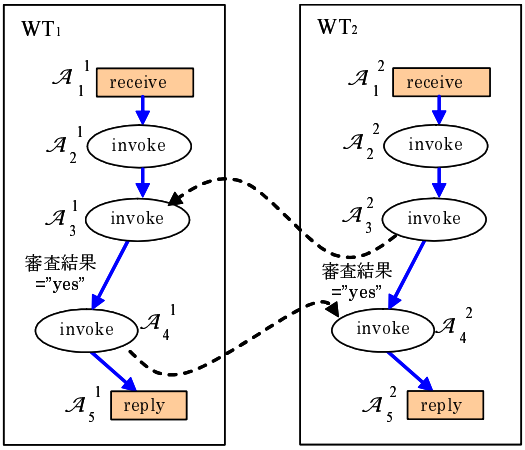

図 4　スケジュール WH_1

3. 隔　離　性

本章では、トランザクションの隔離性を、密結合と疎結合の二つの場合に分けて検討する。

3.1　密結合の場合における隔離性

本節では、密結合 Web サービスのトランザクションの隔離性を、基本的なアクティビティ $a(I, O, M, p)$ に対して入出力データ項目間の対応関係 M についての詳細情報がいっさいない場合の基本モデルと利用できる拡張モデルに分けて検討する。

3.1.1　基本モデルの場合

複合 Web サービスに従う実行であるトランザクションが、複数個同時に実行されるときに単独実行と同様に正当な実行でなければならない。本節では、単独実行時に各トランザクションが満たさなければならない性質を一貫性として形式的に定義し、並行実行時に同じ性質を満たすようにするための隔離性について検討する。

節点の集合であるトランザクションの一貫性は、各節点の一貫性と節点間の一貫性に分けられる。

- 各節点の一貫性は、それぞれの節点の入出力データ間の一貫性を意味する。
- 節点間の一貫性は、各節点とトランザクションのこれまでの実行環境との対応を表す。

基本的なアクティビティは入力データに対して正しく出力データを生成できると仮定するので、原子性の単位となる基本的なアクティビティの実行である節点は、入出力データ間の一貫性を満たす。節点間の一貫性を分析するためには、各節点の作業によってデータベースに対して検索や変更がなされていることを考慮する必要がある。大域変数は並行実行の影響を受けないが、データベースに対する入出力データは例 1 で示したように並行実行の影響を受ける。したがって、大域変数からの入力の場合は、データフローの親の実行環境が並行実行時に受ける影響を再帰的に考慮する必要がある。また、分岐節点は分岐経路上の各節点のデータフロー上の親となるため、分岐条件に使われるデータ項目のデータフロー上の親の実行環境が受けた影響も考慮する必要がある。このため、トランザクションの一貫性は次のように表せる。

[**定義 1**] 入出力データの一貫性を満たす個々の節点からなるトランザクションの一貫性は、トランザクションが単独で実行する場合に、各節点が実行されるときにデータフローの祖先節点のデータベースに対する入出力データ間で満たされている性質を要求する。 □

並行実行において、各々のトランザクションの節点の大域変数が、他のトランザクションの実行によって変更されることはない。このため、一貫性の定義にはそのような入力データとの一貫性を要求していない。WH_1 で示しているように、データベースに対する入出力データはトランザクションの実行中に他のトランザクションによって変更されることによって一貫性上で問題が生じうる。隔離性は、そのような問題に対処するための性質である。

[**定義 2**] 単独実行において一貫性を満たすトランザクション WT_i $(i = 1, \cdots, n)$ からなるスケジュール $WH(HN, HE)$ において、一貫性を満たさないトランザクションが存在しなければ、各々のトランザクションは隔離性を満たす。 □

WH_1 で示したように、WT_2 によって操作されたデータ項目が他のトランザクションによって変更されたため問題が生じた。このため、問題を引き起こしうるすべての変更を禁止することで対処する。

[**定義 3**] トランザクション WT_i $(i = 1, \cdots, n)$ からなるスケジュール $WH(HN, HE)$ において、各々のトランザクションに対して、データベースに対する入出力データが操作されてからデータフローの子孫節点が全て終了するまで他のトランザクションによって変更されていなければ、各々のトランザクションは論理性を満たす。 □

例えば、スケジュール WH_1 においては、$\mathcal{A}_2^2$ のデータベースに対する入出力データ x_3 が、$\mathcal{A}_4^2$ が終了するまで WT_1 中の節点 $\mathcal{A}_4^1$ によって変更されている。このため、WT_2 は論理性を満たさない。

トランザクション WT_i $(i = 1, \cdots, n)$ からなる 2 つのスケジュール WH と WH' で、各トランザクションの実行も最終結果も同じであれば、WH と WH' は等価であるという [2]。

[**定理 1**] 単独実行において一貫性を満たすトランザクション

WT_i $(i = 1, \cdots, n)$ からなるスケジュール $WH(HN, HE)$ において、WH と等価で各 WT_j $(j \in \{1, 2, \cdots, n\})$ が論理性を満たすスケジュール WH' が存在すれば、各 $WT_j (j \in \{1, 2, \cdots, n\})$ は一貫性を満たす。

証明： トランザクション $WT_j(TN_j, TE_j)$ が一貫性を満たさないとすると、定義 1 より節点間の一貫性を満たさない節点 $\mathcal{A}$ が存在することになる。すなわち、$\mathcal{A}$ が実行されるときにデータフローの祖先節点のデータベースに対する入出力データがデータベースの一貫性を満たさない。$\mathcal{A}$ のデータフローの祖先節点のデータベースに対する入出力データを DB' とすると、単独実行時に WT は一貫性を満たすため、WT によって操作された DB' 中のデータ項目が $\mathcal{A}$ が実行されるまで他のトランザクションによって変更されたことになる。しかし、これは各トランザクション WT_j $(j \in \{1, 2, \cdots, n\})$ は WH' において論理性を満たすことと矛盾する。 □

論理性を満たす等価なスケジュールが存在する基準は直列可能性よりも緩和した基準である [12]。施錠方式を用いる場合は、専有施錠と共有施錠間の競合を緩和できることが、論文 [11] によって示されている。また、定理 1 により、各トランザクションが一貫性を満たす単位は、トランザクション単位ではなくデータフローの経路単位でよい。通常トランザクションは複数のデータフローの経路に細分されるため、定理 1 の条件は従来のトランザクション単位での ACID 属性よりも緩和されている。

3.1.2 拡張モデルの場合

定理 1 は、基本モデルにおける隔離性を保証するための一つの十分条件を与えている。本節では、拡張モデルへの適用について検討を行う。

基本モデルにおいては、M についての情報がないため、各節点に対して、各出力データは全ての入力データ項目の関数として扱っていた。一般的に出力データは一部の入力データ項目の関数であるため、入出力データ項目間の詳細な対応関係を記述する M が利用できれば、各データ項目の一貫性上で関連する範囲を詳細化することができる。

[**例 4**] 例 1 中のローン審査アクティビティ $a_3(I_3, O_3, M_3, p_3)$ において、審査結果の x_3 を生成するために経済状況中の借りているローンの総額を参照しない、すなわち、借りているローンの総額と審査結果の x_3 間は対応関係がないとする。この場合、WT_1 の実行中の新たなローンの貸出し作業は WT_1 の一貫性に影響を与えないことになる。すなわち、WH_1 は隔離性を満たすことになる。 □

定理 1 では一貫性を保証するために、操作されたデータベースに対する入出力データ項目がデータフロー上の大域変数によって推移的に関連している全ての子孫が終了するまで他のトランザクションによって変更されないことを要求している。ただし、他のトランザクションの介入はデータフロー上のあらゆる枝に一貫性上で影響を与えるとは限らない。例えば、図 2 で示したように切符予約と発券というデータフローの親子に対しては、切符予約でデータフロー上の子節点である発券の実行をスケジューリングしたため、他のトランザクションの実行によって影響を受けることはなくなる。このため、このような基本的なアクティビティの内部ロジックによって隔離性が保証されたデータフロー上の枝を考慮する必要はない。一般的にデータフローの親子が同一パートナーによって提供された場合には、隔離性について考慮しなくてもよい場合が多い。このような場合には、データフロー上の出力データと入力データの対応関連 M を空にすることによって、そのような枝は考慮されないことになる。

以上の議論から拡張モデルに対する論理性を厳密に定義することができる。

[**定義 4**] トランザクション WT_i $(i = 1, \cdots, n)$ からなるスケジュール $WH(HN, HE)$ において、各々のトランザクションは、データベースに対する入出力データが操作されてから M によって推移的に関連しているデータを持つデータフローの子孫節点が全て終了するまで他のトランザクションによって変更されていなければ、論理性を満たす。 □

明らかに、このように再定義された論理性に対しても定理 1 は成り立つ。

3.2 疎結合の場合における隔離性

疎結合環境におけるトランザクションは、図 3 で示しているように一般的に複数個の密結合環境のサブトランザクションから構成される。

$$a_i(I_i, O_i, M_i, p_i)\ (i = 1, 2, \cdots, k)$$

を基本的なアクティビティとするフローに対して、集合 P に属する企業間の疎結合環境においては、複合 Web サービスのアクティビティ

$$a(\bigcup_{p_i \in P} I_i,\ \bigcup_{p_i \in P} O_i)$$

と表されることになる。すなわち、P 中の要素をパートナーとするアクティビティの入出力メッセージだけが残され、P 以外のパートナー間の入出力データに関する情報は利用できないことになる。例えば、例 1 では、a_1 と a_5 の入出力データが残されることになる。

一般的に、企業内の情報システムを組み立てるための Web サービスは密結合で連携し、企業間の Web サービスは疎結合で連携する。したがって、2 章で述べたように疎結合におけるトランザクションは複数個の密結合で構成されたサブトランザクションから構成される。

各々のサブトランザクションに関して外部入出力データしか得られない疎結合環境は、さらに次の二つの場合に分けられる。

（1） 異なる企業のデータベースに共通部分が存在しない場合。例えば、図 3 で示している旅行代理店、カード会社、航空会社のデータベースの場合である。

（2） 異なる企業のデータベースは共通するデータを扱っている可能性がある場合。例えば、銀行とローン審査会社のデータベースの場合である。

前者の場合は、異なるトランザクションの異なる企業のデータを扱うサブトランザクション間には競合が存在しない。すな

わち、個々の密結合環境におけるサブトランザクションが隔離性を満たせばよい。したがって、本論文の隔離性に関する結果はそのまま疎結合環境におけるサブトランザクションの管理に適用できる。

一方、後者の場合は、個々のサブトランザクションや個々のデータベースに関する詳細な情報がない限り、隔離性については従来の ACID 属性に留まることにならざるをえない。

4. 連携エンジンとデータベースによる実現法

本章では、密結合環境におけるトランザクションの隔離性を保証するための実現法について議論する。従来のデータベースの並行処理制御方式は、他のトランザクションの問題となる実行を禁止する施錠による保守的な方式から時刻印方式のような楽観的な方式まである。本章では基本モデルに対して施錠方式による実現方法について検討する。

4.1 施錠に基づく方式

これまでの議論では各節点 $a(I, O, M, p)$ のデータベースから検索された部分を I_{DB}、データベースへ変更する部分を O_{DB} として記述している。実際の場合には、I_{DB} と O_{DB} に関する情報はデータベースにアクセスするときに初めて分かる。本節では、BPEL4WS エンジンのようなフローエンジンとデータベースの連携による制御方式について検討する。

各々のトランザクションが論理性を満たすスケジュールのみを生成させる制御方式としては、従来の 2 相施錠方式に対して専有施錠と共有施錠間の競合を緩和した拡張 2 相施錠方式が利用できる [11]。すなわち、データ項目に対して専有施錠または共有施錠をした後、再び他のトランザクションによる専有施錠は禁止されるが、共有施錠は禁止されることなくいつでも施錠できる。本論文の基本モデルへ適用すると、次のようになる。

［**定義 5**］ 基本モデルに対する施錠に基づく方式 (Locking protocol) は、次のように施錠と解錠を行う方式である。

- データベース中のデータに対して検索操作／変更操作を行う前に、それぞれ専有施錠／共有施錠を行う。ただし、他のトランザクションによってすでに専有施錠または共有施錠されているデータ項目には再び専有施錠することはできない。
- 大域変数によるデータの引き渡しによって得られるデータフロー情報を活用して、各節点による施錠は、データフロー上の子孫節点がすべて終了するまで保持する。 □

以降、施錠に基づく方式によって生成されるスケジュールを LP スケジュールという。例えば、スケジュール WH_1 において、$\mathcal{A}_2^1$ によって経済状況 x_3 に対する共有施錠は $\mathcal{A}_4^1$ まで保持されることになる。同じように $\mathcal{A}_2^2$ によって経済状況 x_3 に対する共有施錠も $\mathcal{A}_4^2$ まで保持される。したがって、$\mathcal{A}_4^1$ や $\mathcal{A}_4^2$ による x_3 に対する専有施錠は禁止されることになり、WH_1 は LP スケジュールではない。

［**定理 2**］ トランザクション WT_i $(i = 1, \cdots, n)$ からなるスケジュール $WH(HN, HE)$ が LP スケジュールならば、各トランザクション WT_i は基本モデルにおける論理性を満たす。

証明： スケジュール $WH(HN, HE)$ に論理性を満たさないトランザクション WT_i が存在するとする。定義 3 よりデータベースに対する入出力データは、WT_i によって操作された後データフローの子孫節点が全て終了するまでに他のトランザクションによって変更されている。しかし、それは施錠に基づく方式の定義 5 と矛盾する。 □

4.2 改 良 版

施錠に基づく方式の具体的な実現は、データベースに対する検索や変更を行うときに、BPEL4WS エンジンが提供するトランザクションの ID やデータフローに関する情報を利用することによって、定義 5 に従って、施錠、解錠作業や待機させる作業を行うことになる。ただし、WH_1 に施錠に基づく方式を適用すると、すくみが生じることになる。BPEL4WS には例外が発生した場合に補償作業によって対処する仕組みが提供されているので、それを利用してすくみ問題に対処することも考えられるが、Web サービスのトランザクションに対しては、フローが事前に定義されている特徴を利用することができる。本節では、データベース中の各データ項目に対して変更操作を行う前に必ず検索操作を行う場合において、Web サービスの特徴を利用した施錠方式の実現方法について検討する。

フローに従う実行をシミュレーションすることでデータベースに対する操作を行う節点を把握することができる。すなわち、BPEL4WS によって記述されるフローに対して、データベースに対する操作を行う基本的なアクティビティを識別できる。このため、データベースに対して検索操作を行う節点が実行される時点で、その後の実行が同じ部分に対して変更操作を行う可能性があるかどうかの判断はできるようになる。変更操作を行う可能性のある場合は共有施錠でなく専有施錠を行う方式を予約施錠方式という。

［**定義 6**］ 基本モデルに対する予約施錠方式 (Reserve locking protocol) は、次のように施錠、解錠を行う方式である。

- データベース中のデータに対して検索操作／変更操作を行う前に、それぞれ専有施錠／共有施錠を行う。さらに、データベース中のデータに対する検索操作を行うときに、フローの実行によって同じデータ項目を変更する可能性がある場合には、専有施錠を行う。ただし、他のトランザクションによってすでに専有施錠または共有施錠されているデータ項目には再び専有施錠することはできない。
- 大域変数によるデータの引き渡しによって得られるデータフロー情報を活用して、各節点による施錠は、データフロー上の子孫節点がすべて終了するまで保持する。 □

予約施錠方式によって生成されるスケジュールを RLP スケジュールという。例えば、スケジュール WH_1 において、$\mathcal{A}_2^1$ や $\mathcal{A}_2^2$ によって経済状況 x_3 を利用するときには、専有施錠を行うことになる。すでに施錠されているデータ項目に対しては専有施錠できないので、$\mathcal{A}_2^1$ が実行する段階で WT_1 が待機させられる。このため、すくみが生じることはなくなり、WT_2 によってローンを発行した後に WT_1 が実行されることになる。すなわち、WT_1 の情報検索 $\mathcal{A}_2^1$ は最新情報を得ることができ、それに基づいてローン審査 $\mathcal{A}_3^1$ が行われ、審査結果によってそれぞ

れの分岐経路に沿って実行されることになる。

RLP スケジュールは LP スケジュールでもあるので、その中の各トランザクションは基本モデルにおける論理性を満たす。さらに、フローの異なる分岐においてもデータ項目の操作順序が同一であるフローに対しては、次の性質を満たす。

[**定理 3]** 同一フローに従う実行であるトランザクション WT_i $(i = 1, \cdots, n)$ からなるスケジュール $WH(HN, HE)$ が予約施錠方式に従って実行される場合に、すくみが生じることはない。□

証明： 予約施錠方式において、各トランザクションは同一データ項目に対する専有施錠と共有施錠が同時に行われるので、同一データ項目に対する施錠によるすくみが生じえない。さらに、個々のトランザクションは同一フローに従う実行であるので、異なるデータ項目に対する施錠・解錠の順序が同じになる。このため、異なるデータ項目に対する施錠によるすくみも生じえない。 □

5. おわりに

本論文では、インターネット上で公開している Web サービスを部品として組み立てる時代の到来に備えて、Web サービスの連携によって企業内の情報システムを構築する場合に考慮すべき並行処理制御に関する問題について検討を行った。Web サービスに対しては内部ロジックが隠されているため、従来の企業内のワークフローシステムに用いられた内部ロジックによる方法は適用できなくなる。本論文では、Web サービスの連携によって企業内の情報システムを構築する場合のモデルを、疎結合／密結合の場合に分けて与えた。また、密結合のトランザクションに対して、各節点の入出力データを分類することによって、従来の ACID 属性よりも緩和した隔離性を保証するための条件を提案した。さらに、その実現方法は、フローエンジンとデータベースの連携によって行えることを、基本モデルを用いて示した。拡張モデルにおいては、施錠がデータフローの子孫節点まで保持するところを、データフロー中の M によって推移的に関連する子孫節点まで拡張すればよい。それについては今後の課題とする。

[謝辞] 本研究の一部は、文部省科学研究費補助金基盤研究 (C)15500079 の援助を受けている。

文　献

[1] Business Process Execution Language for Web Services (BPEL), Version 1.1, http://dev2dev.bea.com/technologies/ webservices/BPEL4WS.jsp

[2] P. A. Bernstein, V. Hadzilacos and N. Goodman: *Concurrency Control and Recovery in Database Systems*, Addison-Wesley (1987).

[3] Business Transaction Protocol (BTP) http://www.oasis-open.org/committees/ tc_home.php?wg_abbrev=business-transaction

[4] R. Hull and J. Su: Tools for Design of Composite Web Services, ACM SIGMOD Inter. Conf. on Management of Data pp. 958-961 (2004) .

[5] M. Little: Transactions and Web services, Communications of the ACM, Vol. 46, No. 10, pp. 49-54 (2003).

[6] M. P. Papazoglou and J. Dubray: A Survey of Web Service Technologies, Technical Report DIT-04-058, Informatica e Telecomunicazioni, University of Trento (2004).

[7] S. Singh, J. C. Grundy and J. G. Hosking: Developing .NET Web Service-based Applications with Aspect-Oriented Component Engineering, Proc. 5th Autralasian Workshop on Software and Systems Architecures (2004).

[8] Web Service Choreography Interface (WSCI) 1.0, http://www.w3.org/TR/wsci/ (August 2002).

[9] Web Services Transaction (WS-Transaction) http://www-106.ibm.com/developerworks/ webservices/library/ws-transpec/ (August 2002).

[10] 徐海燕，古川哲也：ワークフロートランザクションの隔離性，情報処理学会論文誌:データベース Vol. 44, No. SIG 8 (TOD 18), pp. 55-64 (2003).

[11] 徐海燕，古川哲也，史 一華：並行処理制御方式による独立化可能性クラスと直列可能クラスの比較、情報処理学会論文誌：Vol. 37, No. 8, pp. 1600-1609 (1996).

[12] 徐海燕，古川哲也，史 一華：一貫性情報を用いたデータベースの並行処理制御、情報処理学会論文誌：Vol. 35, No. 12, pp. 2752-2761 (1994).